Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence

Factored Symmetries for Merge-and-Shrink Abstractions

Silvan Sievers and **Martin Wehrle** and **Malte Helmert**
University of Basel, Switzerland
{silvan.sievers,martin.wehrle,malte.helmert}@unibas.ch

Alexander Shleyfman
Technion, Haifa, Israel
alesh@tx.technion.ac.il

Michael Katz
IBM Haifa Research Lab, Israel
katzm@il.ibm.com

Abstract

Merge-and-shrink heuristics crucially rely on effective reduction techniques, such as bisimulation-based shrinking, to avoid the combinatorial explosion of abstractions. We propose the concept of *factored symmetries* for merge-and-shrink abstractions based on the established concept of symmetry reduction for state-space search. We investigate under which conditions factored symmetry reduction yields perfect heuristics and discuss the relationship to bisimulation. We also devise practical merging strategies based on this concept and experimentally validate their utility.

Introduction

Heuristic search is a state-of-the-art approach for optimally solving classical planning problems, and various admissible heuristics have been proposed for this purpose. *Merge-and-shrink heuristics* (Dräger, Finkbeiner, and Podelski 2009; Helmert, Haslum, and Hoffmann 2007; Helmert et al. 2014) are a general class of abstraction-based heuristics. Their main drawback is the costly precomputation phase, which is often a limiting factor for their practical applicability.

To limit the size of intermediate abstractions, *shrinking strategies* are applied during the merge-and-shrink computation. State-of-the-art shrinking strategies are based on (exact or approximate) *bisimulation* (Nissim, Hoffmann, and Helmert 2011). Exact bisimulation can reduce the size of intermediate abstractions without losing precision. A further technique to reduce the size of intermediate abstractions is *label reduction* (Sievers, Wehrle, and Helmert 2014), which unifies transition labels with equivalent behavior.

Complementary to heuristics like merge-and-shrink, *state space pruning techniques* have recently found increasing attention for improving the scalability of optimal search-based planning algorithms. Pruning techniques tackle the state explosion problem by reducing the branching factor of the given planning task. A prominent example is *symmetry reduction*, which was originally proposed for computer-aided verification and has recently seen much interest for planning (Ip and Dill 1996; Pochter, Zohar, and Rosenschein 2011; Domshlak, Katz, and Shleyfman 2012). Symmetry reduction identifies classes of "symmetric" states that do not need to be distinguished and (ideally) only considers one representative of each class.

In this paper, we study the concept of symmetries in the context of merge-and-shrink abstractions. We propose *factored symmetries* for families of abstract transition systems and identify classes of such symmetries for which symmetry reduction on individual abstractions results in perfect merge-and-shrink heuristics. In addition, we investigate and discuss the relationship of local symmetries to the concept of bisimulation and label reduction. Finally, we devise merging strategies maximizing the application of symmetry reduction. Our empirical evaluation within the Fast Downward planning system shows the potential of our approach.

Background

We introduce planning tasks, transition systems, merge-and-shrink heuristics, and bisimulations.

Planning

We consider planning tasks in the SAS$^+$ formalism (Bäckström and Nebel 1995) extended with action costs. A *planning task* is a 4-tuple $\Pi = \langle \mathcal{V}, \mathcal{O}, s_0, s_\star \rangle$, where $\mathcal{V}$ is a finite set of *state variables*, each with a finite domain $dom(v)$. A *partial state* s is a variable assignment over a subset of $\mathcal{V}$, denoted by $vars(s)$. For a partial state s and variable $v \in vars(s)$, we write $s[v]$ for the value assigned to v in s. A partial state s is a *state* if $vars(s) = \mathcal{V}$. The set $\mathcal{O}$ is a finite set of *operators*, each with a *precondition* $pre(o)$, which is a partial state, an *effect* $\mathit{eff}(o)$, also a partial state, and a non-negative *cost* $cost(o) \in \mathbb{R}_0^+$. The state s_0 is the *initial state*, and $s_\star$ is a partial state called the *goal*.

A partial state *complies* with another partial state if they agree on all variables for which both are defined. An operator o is *applicable* in state s if it complies with $pre(o)$. The result of applying o in s is the state s' which complies with $\mathit{eff}(o)$ and satisfies $s'[v] = s[v]$ for all $v \notin vars(\mathit{eff}(o))$.

A *transition system* is a 5-tuple $\Theta = \langle S, L, T, s_0, S_\star \rangle$ where S is a finite set of *states*, L is a finite set of (transition) *labels*, where each label $\ell \in L$ has an associated *cost* $cost(\ell) \in \mathbb{R}_0^+$, $T \subseteq S \times L \times S$ is a set of *labeled transitions*, $s_0 \in S$ is the *initial state*, and $S_\star \subseteq S$ is the set of *goal states*. We write $s \xrightarrow{o} s' \in \Theta$ as a shorthand for $\langle s, o, s' \rangle \in T$. A *plan* $\pi = \langle \ell_1, \ldots, \ell_n \rangle$ for Θ is a path from

the initial state to any goal state, represented by the sequence of labels. Its *cost* is $\sum_{i=1}^{n} cost(\ell_n)$. It is *optimal* if its cost is minimal among all plans.

The *state space* of a planning task Π is the transition system $\Theta^\Pi = \langle S, L, T, s_0, S_\star \rangle$ where S is the set of states of Π, the labels L are the operators of Π with the given costs, $s \xrightarrow{o} s' \in \Theta^\Pi$ whenever operator o is applicable in state s and results in state s', s_0 is the initial state of Π, and $S_\star$ contains all states that comply with the goal of Π. This paper deals with the problem of *optimal planning*, i.e., finding an optimal plan for Θ^Π or showing that no plan exists.

A *heuristic* is a function $h : S \to \mathbb{R}_0^+ \cup \{\infty\}$. The *perfect heuristic* is the heuristic h^* where $h^*(s)$ is the cost of an optimal plan starting in s. A heuristic h is *admissible* if $h(s) \leq h^*(s)$ for all states $s \in S$.

Merge-and-Shrink Heuristics

An *abstraction* of a transition system $\Theta = \langle S, L, T, s_0, S_\star \rangle$ is a surjective function $\alpha : S \mapsto S^\alpha$, where S^α is called the set of *abstract states*. The *abstract transition system* induced by α is the transition system $\Theta^\alpha = \langle S^\alpha, L, T^\alpha, s_0^\alpha, S_\star^\alpha \rangle$ where $T^\alpha = \{\langle \alpha(s), l, \alpha(s') \rangle \mid \langle s, l, s' \rangle \in T\}$, $s_0^\alpha = \alpha(s_0)$ and $S_\star^\alpha = \{\alpha(s) \mid s \in S_\star\}$. The *abstraction heuristic* h^α maps each state $s \in S$ to the cost of an optimal plan in Θ^α starting in abstract state $\alpha(s)$. Abstraction heuristics are always admissible. We will sometimes consider the *induced equivalence relation* $\sim^\alpha$, defined as $s \sim^\alpha t$ iff $\alpha(s) = \alpha(t)$.

Inspired by work in the context of model checking automata networks (Dräger, Finkbeiner, and Podelski 2006), Helmert, Haslum, and Hoffmann (2007) proposed merge-and-shrink heuristics for classical planning. The merge-and-shrink framework allows for fine-grained abstraction heuristics where the abstraction is computed incrementally by iterating between *merging* and *shrinking* steps on sets of abstract transition systems with a common label set. Our exposition is very brief, and we refer to the literature for details (Sievers, Wehrle, and Helmert 2014).

For a planning task $\Pi = \langle \mathcal{V}, \mathcal{O}, s_0, s_\star \rangle$ and a variable $v \in \mathcal{V}$, the *atomic* abstraction α_v is the projection to variable v: $\alpha(s) = s[v]$. The merge-and-shrink computation starts with the set $\mathcal{T} = \{\Theta^{\alpha_v} \mid v \in \mathcal{V}\}$, i.e., with all atomic transition systems. Starting from $\mathcal{T}$, the final abstraction is obtained by repeatedly applying the following transformations:

1. Merging: Two transition systems $\Theta^{\alpha_1}, \Theta^{\alpha_2} \in \mathcal{T}$ are replaced by their synchronized product $\Theta^{\alpha_1} \otimes \Theta^{\alpha_2}$. This corresponds to replacing abstractions α_1 and α_2 by $\alpha_1 \otimes \alpha_2$ defined as $(\alpha_1 \otimes \alpha_2)(s) := (\alpha_1(s), \alpha_2(s))$.
2. Shrinking: A transition system $\Theta^\alpha \in \mathcal{T}$ is replaced by an abstracted version of itself. This corresponds to replacing abstraction α by a coarser abstraction $\beta \circ \alpha$.
3. (Exact) label reduction: let L be the set of labels of $\mathcal{T}$, and let $\ell_1, \ell_2 \in L$ with $cost(\ell_1) = cost(\ell_2)$ be Θ-*combinable* for some $\Theta \in \mathcal{T}$, i.e., for all transition systems $\Theta' \in \mathcal{T} \setminus \{\Theta\}$, ℓ_1 and ℓ_2 are *locally equivalent* (i.e., $s \xrightarrow{\ell_1} s' \in \Theta$ iff $s \xrightarrow{\ell_2} s' \in \Theta$). In all transition systems of $\mathcal{T}$, replace ℓ_1 and ℓ_2 by a fresh label ℓ with the same cost.[1]

A concrete merge-and-shrink heuristic is computed by performing a sequence of such transformations until $\mathcal{T}$ only contains a single transition system, whose corresponding abstraction then defines the resulting heuristic. Merging and exact label reduction preserve information in the sense that, if only they are applied, the resulting heuristic is perfect. This is not generally true for shrinking.

Bisimulation

Bisimulation is a well-known criterion under which an abstract transition system "exhibits the same observable behavior" as the original transition system (Milner 1990).

Nissim, Hoffmann, and Helmert (2011) propose using bisimulation to define shrinking strategies for merge-and-shrink heuristics. Let Θ be a transition system. An equivalence relation $\sim$ on the states of Θ is a *bisimulation* for Θ iff $s \sim t$ implies that *neither* or *both* of s and t are goal states, and for all transitions $s \xrightarrow{\ell} s' \in \Theta$ there exists a transition $t \xrightarrow{\ell} t'$ where $s' \sim t'$. (These rules are perhaps easiest to understand with the intuition that states that are *not* bisimilar are ones that "behave differently" in an important way. If one of the states s and t is a goal state and the other one is not, then they certainly behave differently. Similarly, if there is a transition label that takes us from s and t to states that behave differently – or that labels an outgoing transition in only one of the two states – then they also behave differently.) An abstraction α is a bisimulation if the induced equivalence relation $\sim^\alpha$ is. Shrinking with a bisimulation is information-preserving.

Abstraction-Based Symmetries

Symmetry reduction for state-space search is based on *automorphisms* (symmetries) of the state space. The set of all such automorphisms is closed under function composition and defines the *automorphism group* of the state space. Each subgroup of the automorphism group induces an equivalence relation on the state space, and only one state in each equivalence class needs to be considered during search.

Current approaches find symmetries based on factored planning task representations called *problem description graphs*, or *PDGs* for short (Pochter, Zohar, and Rosenschein 2011). We generalize this concept to *factored symmetries* based on arbitrary sets of abstract transition systems, study the special cases of *local* and *atomic* symmetries and explore their relationship to bisimulation and label reduction.

Factored Symmetries

We begin by introducing *factored symmetries*, which are defined on sets of transition systems $\mathcal{T}$ with a common label set. To simplify notation, we assume throughout the paper that states of different transition systems in $\mathcal{T}$ can always be distinguished, i.e., if S^i and S^j are state sets of different transition systems in $\mathcal{T}$, then $S^i \cap S^j = \emptyset$. If σ is a function defined on states and labels, we extend it in the natural way to sets of states ($\sigma(X) = \{\sigma(x) \mid x \in X\}$), sets of sets of states ($\sigma(\mathcal{X}) = \{\sigma(X) \mid X \in \mathcal{X}\}$) and transitions ($\sigma(\langle s, \ell, s' \rangle) = \langle \sigma(s), \sigma(\ell), \sigma(s') \rangle$).

[1] There exists a second kind of exact label reduction based on global subsumption of labels, which is not relevant for this paper.

Definition 1. *Let $\mathcal{T} = \{\Theta^1, \ldots, \Theta^n\}$ be a set of transition systems with common label set L, where $\Theta^i = \langle S^i, L, T^i, s_0^i, S_\star^i \rangle$ for all $i \in \{1, \ldots, n\}$. A* factored symmetry *of $\mathcal{T}$ is a permutation σ of the set $\bigcup_{i=1}^n S^i \cup L$ that maps states to states and labels to labels such that*

1. $\sigma(\{S^1, \ldots, S^n\}) = \{S^1, \ldots, S^n\}$
2. $\sigma(\bigcup_{i=1}^n S_\star^i) = \bigcup_{i=1}^n S_\star^i$
3. $\sigma(\bigcup_{i=1}^n T^i) = \bigcup_{i=1}^n T^i$
4. $cost(\sigma(\ell)) = cost(\ell)$ *for all* $\ell \in L$

The four conditions guarantee that a factored symmetry preserves the grouping of states into transition systems, the goal state property, the transition structure, and label costs. We illustrate the definition with the example in Figure 1, corresponding to a planning task with four variables a, b, c and d and uniform-cost operators $o_{x,y}$ for certain pairs of $x, y \in \{a, b, c, g\}$ that change x from 1 to 0 and y from 0 to 1. The mapping σ that cyclically rotates all facts and operators related to variables $\{a, b, c\}$ is a factored symmetry.

When applied to the atomic transition systems of a planning task, Definition 1 is equivalent to the definition of PDG symmetries in earlier work. However, it is not limited to this case. On the opposite end of the spectrum, the case $\mathcal{T} = \{\Theta^\Pi\}$ captures a semantic notion of state space symmetry in a non-factored representation.

More generally, the definition can be applied to any intermediate result in the computation of a merge-and-shrink abstraction. We say that a set of transition systems $\mathcal{T}$ *represents* the transition system $\mathcal{T}^\otimes$ obtained by merging its components. A factored symmetry σ of $\mathcal{T}$ naturally induces a permutation $\sigma^\otimes$ on the states $S^\otimes$ of $\mathcal{T}^\otimes$, and Definition 1 guarantees that $h^*(\sigma^\otimes(s)) = h^*(s)$ for all $s \in S^\otimes$.

Like other notions of symmetry, factored symmetry is closed under composition (if σ and τ are factored symmetries, then so is $\sigma \circ \tau$) and hence induces a group structure. If Γ is a set of factored symmetries for $\mathcal{T}$, then we say that two states s and s' of $\mathcal{T}^\otimes$ are *symmetric under* Γ, in symbols $s \sim^\Gamma s'$, if $\sigma^\otimes(s) = s'$ for some factored symmetry σ in the group generated by Γ. This is an equivalence relation, and states in the same equivalence class always share the same h^* value. The factored symmetry in Figure 1 induces an equivalence relation showing that, for example, all states containing g_0 and *exactly one* of $\{a_1, b_1, c_1\}$ are symmetric.

In the special case where Γ is the set of all factored symmetries of $\mathcal{T}$, we simplify this notation to $s \sim s'$ and say that s and s' are *symmetric under factored symmetry*.

Interaction with Merging and Shrinking

We want to combine reasoning about symmetries with the manipulation of transition systems in the merge-and-shrink framework. It is then natural to ask how factored symmetries interact with merging and shrinking.

It is easy to see that the interaction between *shrinking* and factored symmetries is in general unpredictable. For example, we may have $s \sim s'$ in the set of transition systems $\mathcal{T}$, but $\beta(s) \not\sim \beta(s')$ for the corresponding abstracted states after shrinking with abstraction β. This happens when a shrinking step "breaks" an existing symmetry, for example by combining a_0 and a_1 in Figure 1. The converse case is also possible: we may have $s \not\sim s'$ before shrinking, but $\beta(s) \sim \beta(s')$ after shrinking if the abstraction removes the obstacles to the symmetry of s and s'. This should come as little surprise because shrinking can transform a transition system in essentially arbitrary ways. We will discuss the special case of bisimulation-based shrinking later.

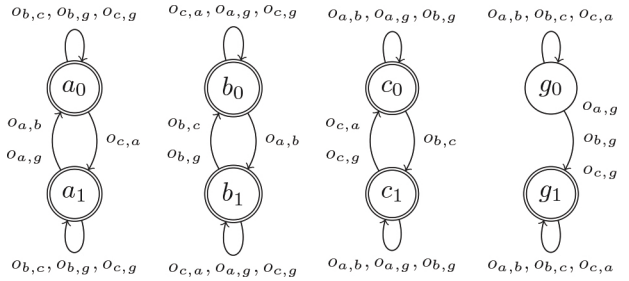

Figure 1: Factored symmetry example: rotating $a \mapsto b \mapsto c \mapsto a$ in all abstract states (e.g., $\sigma(b_1) = c_1$) and labels (e.g., $\sigma(o_{a,b}) = o_{b,c}$) is a factored symmetry.

Perhaps somewhat surprisingly, *merging* can also affect the symmetry properties of $\mathcal{T}$ unpredictably. To see this, observe that the symmetry property between states $a_1 b_0 c_0 g_0$ and $a_0 b_0 c_1 g_0$ is lost when merging the two transition systems at the left of Figure 1 (which shows that symmetry can be lost by merging), and the same symmetry is recovered by then merging this product by the third transition system (showing that symmetries can be gained by merging). In summary, these considerations show that it can be beneficial to search for new symmetries after each transformation step in the construction of merge-and-shrink heuristics.

Local and Atomic Symmetries

Non-factored state spaces have the property that combining symmetric states (formally: abstracting by mapping each state to its orbit in the symmetry group) preserves optimal goal distances. It is thus natural to attempt using *factored symmetries* for information-preserving shrinking in the merge-and-shrink framework. The first obstacle to this is that shrinking happens at the level of individual transition systems $\Theta \in \mathcal{T}$, while symmetries like the one in Figure 1 can critically rely on relationships between different transition systems. We thus now consider a subclass of symmetries that "stay within" the individual abstractions.

Definition 2. *A factored symmetry σ of a set of transition systems $\mathcal{T}$* stabilizes *$\Theta \in \mathcal{T}$ if σ maps states of Θ to states of Θ. A factored symmetry is* local *if it stabilizes all $\Theta \in \mathcal{T}$.*

Local symmetries are closed under composition and hence form a subgroup of the group of factored symmetries. Since local symmetries stay within each abstraction in $\mathcal{T}$, they can be used to define an equivalence class on *abstract states*: for all $\Theta \in \mathcal{T}$ and all states s, s' of Θ, we set $s \sim s'$ if there exists a local symmetry σ with $\sigma(s) = s'$.

By analogy to the use of symmetries in the global state space, it may appear natural to use this equivalence relation as a basis for abstraction (shrinking). However, such a use of

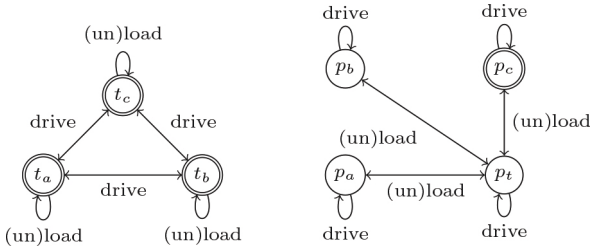

Figure 2: Example transportation task: atomic abstractions Θ^t of truck (left) and Θ^p of package (right). For brevity, edges generally correspond to several labels.

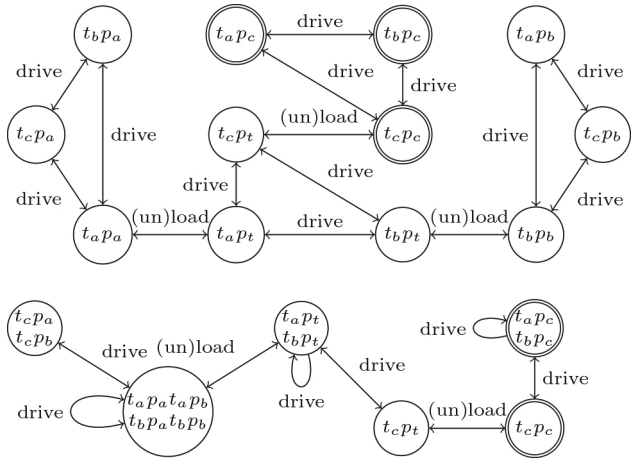

Figure 3: Example transportation task: synchronized product $\Theta^t \otimes \Theta^p$ (above) and abstraction obtained by shrinking based on local symmetries before merging (below). For brevity, edges generally correspond to several labels.

symmetries is *not* information-preserving, i.e., can lead to a loss of precision in the resulting merge-and-shrink heuristic.

Proposition 1. *Shrinking based on local symmetries is not information-preserving.*

To prove the proposition, we give an example where shrinking based on local symmetries before merging results in an imperfect heuristic. We use a simple transportation task Π with a truck t, a package p, and locations a, b and c. Figure 2 shows the abstract transition systems Θ^t and Θ^p for the projections to the truck and package. All operator costs are 1. As indicated by the double circles, the goal is for the package to be at c. The initial state can be chosen arbitrarily.

There is a local symmetry σ that swaps the role of locations a and b everywhere (e.g., $\sigma(t_a) = t_b$; $\sigma(p_a) = p_b$). Shrinking based on local symmetries would thus combine t_a and t_b to a common abstract state ("truck at a or b"), and similarly it would combine p_a and p_b ("package at a or b").

Figure 3 shows the unabstracted global state space (top) compared to the transition system obtained by performing this abstraction and then merging (bottom). We see that the abstraction loses precision: for example, it leads to the estimate $h(t_b p_a) = 3$ instead of the correct $h^*(t_b p_a) = 4$. Intuitively, even though locations a and b are *locally* symmetric in Θ^t and Θ^p, they cannot simply be combined: this loses the distinction between states where the truck and package are at the same location and those where they are not.

This problem can be resolved by further restricting the notion of local symmetries to *atomic* symmetries.

Definition 3. *A factored symmetry σ of a set of transition systems $\mathcal{T}$* affects *$\Theta \in \mathcal{T}$ if there exists a state s of Θ with $\sigma(s) \neq s$. A factored symmetry is* atomic *if it affects at most one transition system $\Theta \in \mathcal{T}$.*

Clearly atomic symmetries are a subset of local symmetries. Unlike local symmetries, they do not form a group: the composition of atomic σ_1 and σ_2 is non-atomic if they affect different transition systems. However, for any *fixed* Θ, the atomic symmetries affecting at most Θ form a group.

Proposition 2. *Shrinking based on atomic symmetries is information-preserving.*

In more detail, let $\mathcal{T} = \{\Theta_1, \ldots, \Theta_n\}$, and let $\mathcal{T}' = \{\Theta'_1, \ldots, \Theta'_n\}$, where each Θ'_i is obtained from Θ_i by abstracting based on an atomic symmetry affecting (at most) Θ_i. Then $\mathcal{T}'^{\otimes}$ defines a perfect heuristic for $\mathcal{T}^{\otimes}$. We postpone the proof of this proposition, as it follows from a more general result in the following section.

That fact that they allow for shrinking strategies that maintain perfection makes atomic symmetries particularly attractive, and we would like to exploit this result even for abstractions that are not atomic. The following result shows that we can achieve this by merging all transition systems affected by a given symmetry.

Proposition 3. *Let σ be a factored symmetry of a set of transition systems $\mathcal{T}$ that affects (exactly) the transition systems $\mathcal{T}^\sigma \subseteq \mathcal{T}$. Let $\mathcal{T}'$ be the set of transition systems obtained from $\mathcal{T}$ by merging all transition systems in $\mathcal{T}^\sigma$. Then $\mathcal{T}'$ contains an atomic symmetry σ' that induces the same symmetry on $\mathcal{T}^\otimes$ as σ.*

Proof. Let $\mathcal{T} = \{\Theta_1, \ldots, \Theta_n\}$, and let $k \in \{1, \ldots, n\}$ such that σ affects exactly the transition systems Θ^i with $i \leq k$. We get $\mathcal{T}' = \{\Theta', \Theta_{k+1}, \ldots, \Theta_n\}$ with $\Theta' = \bigotimes_{i=1}^{k} \Theta_i$. States of Θ' can be written as sets of the form $\{s_1, \ldots, s_k\}$, where each s_i is a state of Θ_i. (It is more common to use tuples instead of sets, but the two representations are equivalent because we require states of different transition systems to be disjoint. Using sets simplifies the definition of σ'.)

We define $\sigma'(\ell) = \sigma(\ell)$ for all labels ℓ, $\sigma'(s_j) = s_j$ for all states s_j of $\Theta_{k+1}, \ldots, \Theta_n$, and finally $\sigma'(\{s_1, \ldots, s_k\}) = \{\sigma(s_1), \ldots, \sigma(s_k)\}$. It is easy to verify that σ' satisfies all properties of symmetries (because σ does) and that it induces the same symmetry on $\mathcal{T}^\otimes$ as σ. Also, it is clearly atomic, affecting only Θ'. □

Relationship to Bisimulation

Symmetry and bisimulation are similar concepts, as both identify (possibly smaller) structures with equivalent behavior. Hence, a natural question is to ask about their relationship. For non-factored transition systems, it is well-known

that every symmetry induces a bisimulation (Clarke, Grumberg, and Peled 2000). It is also easy to see that the converse does not hold. The transportation task example shows that nontrivial factored symmetries, even local ones, do not necessarily induce nontrivial bisimulations on the individual transition systems. (There are no nontrivial bisimulations in the abstract transition systems of the example.)

We now show that for *fully label-reduced* sets of transition systems (i.e., where no two labels can be combined by exact label reduction), *atomic* symmetries are captured by bisimulation.

Proposition 4. *Let $\mathcal{T}$ be a fully label-reduced set of transition systems, and let $\Theta \in \mathcal{T}$. Let $\sim$ be the equivalence relation on Θ induced by the atomic symmetries of $\mathcal{T}$ affecting at most Θ. Then $\sim$ is a bisimulation of Θ.*

Proof. Let S be the states of Θ. We show that $\sim \subseteq S \times S$ satisfies the two properties of bisimulations: (1) if $s \sim t$, then neither or both of s and t are goal states; (2) if $s \sim t$ and $s \xrightarrow{\ell} s' \in \Theta$, then $t \xrightarrow{\ell} t' \in \Theta$ for some t' with $s' \sim t'$.

Consider $s, t \in S$ with $s \sim t$. Then there exists a local symmetry σ of $\mathcal{T}$ affecting at most Θ such that $\sigma(s) = t$. From the goal-perserving property of symmetries, we get (1). To show (2), consider $\ell \in L$ and $s' \in S$ such that $s \xrightarrow{\ell} s' \in \Theta$. From the definition of symmetry, we get $\sigma(s) \xrightarrow{\sigma(\ell)} \sigma(s') \in \Theta$. Defining $t' = \sigma(s')$, we obtain $t \xrightarrow{\sigma(\ell)} t' \in \Theta$ and $s' \sim t'$. To complete the proof, we will show $\sigma(\ell) = \ell$ and hence $t \xrightarrow{\ell} t' \in \Theta$, which proves (2).

Consider any of the *other* transition systems $\hat{\Theta} \in \mathcal{T} \backslash \{\Theta\}$. Because σ is an atomic symmetry affecting at most Θ, it maps each state of $\hat{\Theta}$ to itself. Hence, σ maps each $\hat{s} \xrightarrow{\ell} \hat{s}' \in \hat{\Theta}$ to $\sigma(\hat{s}) \xrightarrow{\sigma(\ell)} \sigma(\hat{s}') = \hat{s} \xrightarrow{\sigma(\ell)} \hat{s}' \in \hat{\Theta}$. In other words, every transition with label ℓ in $\hat{\Theta}$ has a parallel transition with label $\sigma(\ell)$. The converse also holds because σ^{-1} is also an atomic symmetry affecting at most Θ. This shows that ℓ and $\sigma(\ell)$ are locally equivalent in $\hat{\Theta}$.

This local equivalence holds for all $\hat{\Theta} \neq \Theta$, and hence ℓ and $\sigma(\ell)$ are Θ-combinable. Moreover, we have $cost(\sigma(\ell)) = cost(\ell)$ because σ is a factored symmetry. Therefore, if $\sigma(\ell) \neq \ell$, the two labels satisfy the two requirements for exact label reduction, which contradicts our requirement that $\mathcal{T}$ is fully label-reduced. Hence we must have $\sigma(\ell) = \ell$, concluding the proof. □

Proposition 2 follows as a corollary because shrinking based on bisimulation is known to be information-preserving (Helmert et al. 2014). More importantly, the result shows that existing bisimulation-based shrink strategies *automatically* capture redundancies exploitable by atomic symmetries when using full label reduction.

Experiments

In this section, we propose and evaluate a general approach for including symmetry reasoning in the merge-and-shrink framework. We have seen that shrinking based on atomic symmetries is information-preserving, while shrinking based on other classes of factored symmetries is not. We exploit this information by tailoring the *merging strategy* to force the occurrence of atomic symmetries. Specifically, if we detect a factored symmetry σ affecting k transition systems and then merge these transition systems, σ becomes an atomic symmetry only affecting the merged system.

We have also seen that state-of-the-art approaches using bisimulation-based shrinking and label reduction based on Θ-combinability automatically exploit atomic symmetries, so there is no need to adapt the shrinking strategy.

Finally, we have seen that new symmetries can arise at any time during the merge-and-shrink process, so it can pay off to search for new factored symmetry in every iteration of the merge-and-shrink loop.

Algorithm 1 Symmetry-based merge-and-shrink.

```
1  If |N| ≤ 1:
2      Compute a set Σ of non-atomic symmetries of T.
3      If Σ ≠ ∅:
4          Let N := {Θ ∈ T | Θ is affected by one chosen σ ∈ Σ_n}
5  If |N| ≥ 2:
6      Choose Θ_1, Θ_2 ∈ N.
7  else
8      Choose Θ_1, Θ_2 according to basic merging strategy M.
9  Apply label reduction w.r.t. basic strategy L on all abstractions in T.
10 Apply shrinking w.r.t basic strategy S on Θ_1, Θ_2.
11 Replace Θ_1, Θ_2 with Θ_1 ⊗ Θ_2 in N (if applicable) and T.
```

Algorithm 1 shows one iteration of the symmetry-enhanced merge-and-shrink algorithm resulting from these observations. It augments an existing merging strategy M, shrinking strategy S and label reduction strategy L as follows: initially, let $N := \emptyset$. As usual, the merge-and-shrink procedure is executed until $|\mathcal{T}| = 1$. Whenever we are currently not pursuing any merging policy based on symmetries (line 1 triggers), we compute factored symmetries of $\mathcal{T}$ and, if any are found, store all affected abstractions of a chosen one in N (line 4). The next pair of abstractions to be merged is chosen either according to the set N, if it contains at least two abstractions (line 6), or according to M (line 8). The rest of the procedure corresponds to regular merge-and-shrink.

Two design choices remain: the choice of one non-atomic symmetry σ of the set Σ (line 4) and the decision which pair of transition systems to choose from N (line 6). Regarding the first choice, we tried selecting the symmetry which affects the least or the most abstractions of $\mathcal{T}$. The latter led to slightly better results in preliminary experiments, and we report results for this configuration. Concerning the second choice, we settled on the following non-linear policy: we first merge all abstractions which are non-locally affected (i.e., mapped onto other abstractions) separately for each such cycle of mapped abstractions. After these merges, the induced symmetry is local, and we linearly merge all remaining abstractions affected by the symmetry.

Experimental Setup

We evaluate our approach for optimal planning with A* and the merge-and-shrink framework in the Fast Downward planner (Helmert 2006), using all optimal IPC benchmarks up to 2011 that supported by merge-and-shrink (44 domains

	Coverage (blind search: 519)			#successful M&S constr.		
	base	symm	symm-1	base	symm	symm-1
CGGL-B-N50K	600	**646**	637	1138	1177	**1181**
DFP-B-N50K	644	**657**	646	1181	**1204**	1203
MIASM-B-N50K	654	659	**660**	1159	1162	**1172**
RL-B-N50K	634	**652**	643	1202	**1219**	1216
RND-B-N50K	583	**622**	605	1165	**1207**	1200

Table 1: Coverage (left part) and number of M&S abstractions built within the resource limits (right part) for different M&S strategies (see section "Experimental Setup" for abbreviations). *Base*: original strategy; *symm*: strategy with our symmetry enhancement; *symm-1*: strategy with our symmetry enhancement limited to symmetries found on the set of atomic abstractions in the first M&S iteration. Best configuration for every M&S strategy in bold.

and 1396 instances). Our experiments are performed on machines with Intel Xeon E5-2660 CPUs running at 2.2 GHz, using a time bound of 30 minutes and a memory bound of 2 GB per run. Factored symmetries are computed with Bliss (Junttila and Kaski 2007). We limit the overall time budget for Bliss to $T = 60$ seconds, skipping line 1 in Algorithm 1 once this time budget is exhausted.

We use state-of-the-art strategies for merge-and-shrink, including the linear merging strategies *reverse level RL* (Nissim, Hoffmann, and Helmert 2011), *causal graph goal level CGGL* (Helmert, Haslum, and Hoffmann 2007) and the *random* strategy *RND* based on a random variable order, and the non-linear merging strategies *DFP* (Sievers, Wehrle, and Helmert 2014; Dräger, Finkbeiner, and Podelski 2009) and *MIASM* (Fan, Müller, and Holte 2014).[2] The most powerful shrinking strategies are based on bisimulations (Helmert et al. 2014; Katz, Hoffmann, and Helmert 2012). We focus on the shrinking strategy based on *bisimulation B* with limit $N = 50000$ for the maximal size of all transition systems during the merge-and-shrink computation.

Experimental Results

Table 1 shows the summary of our results. Most importantly, we observe that the symmetry enhancement (*symm*) increase coverage for all strategies. Apparently, one reason for this is the ability to successfully finish the computation of the merge-and-shrink abstractions more often. Clearly, being able to successfully compute the merge-and-shrink abstraction is an obvious, yet important requirement of successfully finding plans with the resulting heuristic. The number of successful M&S constructions is shown on the right part of Table 1, demonstrating a significant increase compared to the baseline strategies (*base*).

[2]While merging for symmetries seamlessly fits the linear merging strategies (whenever the atomic abstraction to be merged next according to the linear merging strategy has been merged with some other abstractions, we simply take the result of the previous merge(s)) and also the dynamically computed DFP strategy, this is not true for the MIASM strategy, which is precomputed before the actual merge-and-shrink computation. Our current implementation of MIASM does *not* take into account symmetries during the precomputation step, and hence we arbitrarily "break" the precomputed merging order when merging according to symmetries in intermediate merge-and-shrink iterations.

Coverage	CGGL base	CGGL symm
gripper (20)	7	**+11**
parking-opt11-strips (20)	0	**+7**
mystery (30)	12	**+5**
pipesworld-tankage (50)	9	**+5**
airport (50)	11	**+4**
miconic (150)	74	**+4**
mprime (35)	20	**+3**
pipesworld-notankage (50)	12	**+3**
sokoban-opt08-strips (30)	27	**+3**
elevators-opt08-strips (30)	12	**+1**
elevators-opt11-strips (20)	10	**+1**
trucks-strips (30)	7	**+1**
visitall-opt11-strips (20)	9	**+1**
woodworking-opt08-strips (30)	11	**+1**
woodworking-opt11-strips (20)	6	**+1**
logistics98 (35)	**5**	-1
nomystery-opt11-strips (20)	**19**	-1
satellite (36)	**7**	-1
tidybot-opt11-strips (20)	**1**	-1
zenotravel (20)	**11**	-1
Sum (716)	270	**+46**
Remaining domains (680)	**330**	**±0**
Sum (1396)	600	**646**

Table 2: Improvement in coverage on a per-domain basis for CGGL base vs. CGGL symm. Best values in bold.

To show the benefits of recomputing symmetries in every merge-and-shrink iteration, Table 1 also shows results where Bliss is only invoked in the first merge-and-shrink iteration (*symm-1*). Clearly, running Bliss only once is beneficial already, but the overall strategy profits from the information gained in further runs (with the exception of MIASM).

The strongest benefits with our framework are obtained for the CGGL strategy with 46 additional problems solved. Considering the usual exponential growth in complexity in the size of the problem instances, this is a substantial improvement. The lowest coverage increase is obtained for the MIASM strategy. Again, note that our integration of MIASM and symmetries is very basic (see footnote 2 and section "Conclusions"). We further investigate per-domain performance of the CGGL strategy to verify that the symmetry enhancement is beneficial in a broad range of domains. Table 2 shows that coverage improves in 15 out of 44 domains, whereas it decreases in 5 domains, and then only by 1 task. In the remaining 24 domains, the configurations achieve equal coverage and hence the domains are displayed in an aggregated row.

Considering the quality of the resulting heuristic, we again provide details for the CGGL strategy. Figure 4 compares the number of expansions of CGGL base against CGGL symm on all domains. Figure 5 shows the same data in a way that allows distinguishing the different domains,

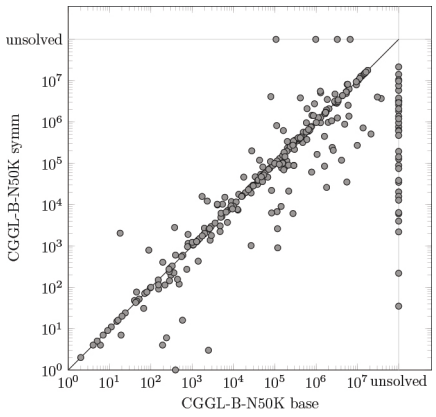

Figure 4: Expansions for CGGL base vs. CGGL symm.

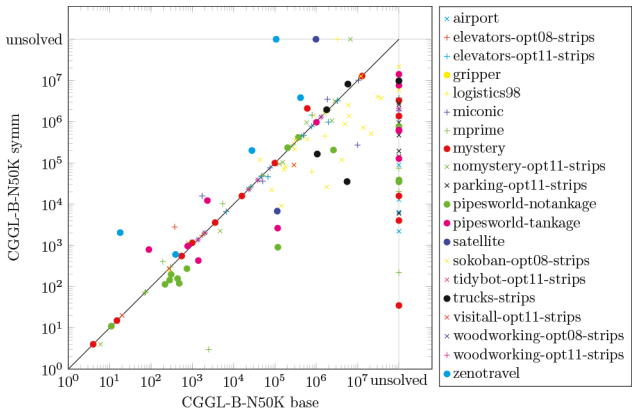

Figure 5: Expansions for CGGL base vs. CGGL symm only on domains where both configurations do *not* obtain the same coverage.

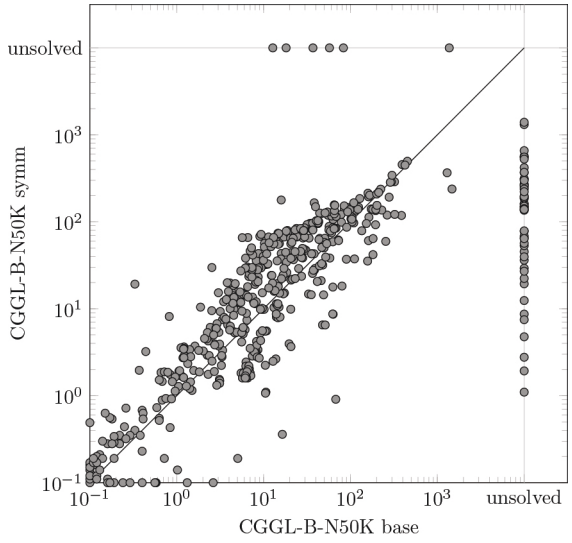

Figure 6: Runtime for CGGL base vs. CGGL symm.

Outcome	CGGL base	CGGL symm	CGGL symm-1
M&S out of memory	143	100	**96**
M&S out of time	**115**	119	119
Search out of memory	530	**524**	536
Search out of time	4	**2**	3
Proved unsolvable	4	**5**	**5**
Solved	600	**646**	637

Table 3: Detailed outcomes (such as reasons for failing to solve a task) for all three CGGL configurations. Best values in bold.

Outcome	CGGL base vs. symm	CGGL base vs. symm-1
M&S out of memory	27	23
M&S out of time	0	0
Search out of memory	25	19
Search out of time	0	0
Solved	594	595

Table 4: Behavior of CGGL base on the tasks solved by the two symmetry-enhanced strategies. Column CGGL base vs. symm shows results only for the tasks solved by CGGL base; similarly for the other column.

for clarity restricted to the domains of Table 2 where the two configurations do *not* achieve the same coverage. Finally, we also compare the runtime of both configurations on the full benchmark set (Figure 6). While there is a larger number of cases where the symmetry-based configuration requires significantly fewer expansions than for the baseline, the clearest distinguishing characteristic is that there are far more tasks that the symmetry-based configuration solves but the baseline does not.

We hence further investigate the reasons of failure for the three CGGL configurations. Table 3 lists the number of tasks for which either the merge-and-shrink construction or the search runs out of memory or time, as well as the number of solved tasks and tasks proved unsolvable. We observe that using symmetries drastically decreases the number of tasks where the configuration runs out of memory, which means that we obtain more compact abstractions with the improved merging strategy. At the same time, the number of timeouts remains nearly the same both for the merge-and-shrink computation and the search.

To see the differences between the CGGL baseline and our two symmetry-enhanced strategies, we report the same reasons of failure for the CGGL baseline on benchmarks solved by the symmetry-enhanced CGGL configuration we compare against. Table 4 shows the comparison of CGGL base against CGGL symm (left column) and against CGGL symm-1 (right column). We observe that all 52 tasks that the baseline cannot solve compared to CGGL symm are due to reaching the memory limit, where in 27 cases this limit is reached during the merge-and-shrink computation. Similar results hold when comparing against CGGL symm-1. These observations match the previous ones.

Bliss Failures	CGGL symm	CGGL symm-1
Bliss out of memory	145	**2**
Bliss out of time	386	**0**

Table 5: Number of tasks for which the computation of symmetries with Bliss failed for CGGL configurations.

To show the impact on resource consumption of using Bliss for symmetry detection, Table 5 lists the number of tasks for which the symmetry computation runs out of memory or time, if the time limit applies (configuration CGGL symm). Note that whenever Bliss runs out of memory or time, we continue the merge-and-shrink computation without any further use of symmetries. We observe that symmetry computation is expensive and often reaches resource limits. Our previously discussed results show, however, that computing symmetries either only once or for a relatively short total time of one minute pays off.

Conclusions

We introduced factored symmetries for sets of transition systems and discussed their relationship to the merge-and-shrink framework. We showed that merging and shrinking operations can lead to the loss of symmetries as well as the discovery of new ones and that the special class of atomic symmetries is captured by bisimulation-based shrinking and exact label reduction. We proposed a general way to enhance merge-and-shrink computations by symmetry reasoning and experimentally demonstrated its utility.

One possible future research direction is the tighter integration of symmetry reasoning with the MIASM strategy, which is based on estimating the required size of intermediate abstractions. Symmetry information could be used to make this estimation process more accurate. Another direction for future work is to study imperfect shrinking strategies using symmetry information, for example based on non-atomic local symmetries.

Acknowledgments

We thank the anonymous reviewers for their comments, which helped improve the paper. This work was supported by the Swiss National Science Foundation (SNSF) and by the Israel Science Foundation (ISF) grant 1045/12.

References

Bäckström, C., and Nebel, B. 1995. Complexity results for SAS^+ planning. *Computational Intelligence* 11(4):625–655.

Clarke, E. M.; Grumberg, O.; and Peled, D. A. 2000. *Model Checking*. The MIT Press.

Domshlak, C.; Katz, M.; and Shleyfman, A. 2012. Enhanced symmetry breaking in cost-optimal planning as forward search. In McCluskey, L.; Williams, B.; Silva, J. R.; and Bonet, B., eds., *Proceedings of the Twenty-Second International Conference on Automated Planning and Scheduling (ICAPS 2012)*. AAAI Press.

Dräger, K.; Finkbeiner, B.; and Podelski, A. 2006. Directed model checking with distance-preserving abstractions. In Valmari, A., ed., *Proceedings of the 13th International SPIN Workshop (SPIN 2006)*, volume 3925 of *Lecture Notes in Computer Science*, 19–34. Springer-Verlag.

Dräger, K.; Finkbeiner, B.; and Podelski, A. 2009. Directed model checking with distance-preserving abstractions. *International Journal on Software Tools for Technology Transfer* 11(1):27–37.

Fan, G.; Müller, M.; and Holte, R. 2014. Non-linear merging strategies for merge-and-shrink based on variable interactions. In *Proceedings of the Seventh Annual Symposium on Combinatorial Search (SoCS 2014)*, 53–61. AAAI Press.

Helmert, M.; Haslum, P.; Hoffmann, J.; and Nissim, R. 2014. Merge-and-shrink abstraction: A method for generating lower bounds in factored state spaces. *Journal of the ACM* 61(3):16:1–63.

Helmert, M.; Haslum, P.; and Hoffmann, J. 2007. Flexible abstraction heuristics for optimal sequential planning. In Boddy, M.; Fox, M.; and Thiébaux, S., eds., *Proceedings of the Seventeenth International Conference on Automated Planning and Scheduling (ICAPS 2007)*, 176–183. AAAI Press.

Helmert, M. 2006. The Fast Downward planning system. *Journal of Artificial Intelligence Research* 26:191–246.

Ip, C. N., and Dill, D. L. 1996. Better verification through symmetry. *Formal Methods in System Design* 9(1–2):41–75.

Junttila, T., and Kaski, P. 2007. Engineering an efficient canonical labeling tool for large and sparse graphs. In *Proceedings of the Ninth Workshop on Algorithm Engineering and Experiments (ALENEX 2007)*, 135–149. SIAM.

Katz, M.; Hoffmann, J.; and Helmert, M. 2012. How to relax a bisimulation? In McCluskey, L.; Williams, B.; Silva, J. R.; and Bonet, B., eds., *Proceedings of the Twenty-Second International Conference on Automated Planning and Scheduling (ICAPS 2012)*, 101–109. AAAI Press.

Milner, R. 1990. Operational and algebraic semantics of concurrent processes. In van Leeuwen, J., ed., *Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics*. Elsevier and MIT Press. 1201–1242.

Nissim, R.; Hoffmann, J.; and Helmert, M. 2011. Computing perfect heuristics in polynomial time: On bisimulation and merge-and-shrink abstraction in optimal planning. In Walsh, T., ed., *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI 2011)*, 1983–1990.

Pochter, N.; Zohar, A.; and Rosenschein, J. S. 2011. Exploiting problem symmetries in state-based planners. In Burgard, W., and Roth, D., eds., *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence (AAAI 2011)*, 1004–1009. AAAI Press.

Sievers, S.; Wehrle, M.; and Helmert, M. 2014. Generalized label reduction for merge-and-shrink heuristics. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence (AAAI 2014)*, 2358–2366. AAAI Press.